Army Research Laboratory

Voting Techniques for Combining Multiple Classifiers

by Diana Thomas and Sandor Z. Der

ARL-TR-1549

March 1998

Abstract

Algorithms for decision fusion are surveyed and qualitatively compared for the problem of classification of targets. The methods assume that a number of imperfect classifiers are available, and that these classifiers have enough statistical independence that significant improvement can be made by good combination algorithms. Optimal solutions for this problem require an exact statistical model of the classifiers and the decision space, which are rarely available for real-world problems. Consequently, algorithms must be chosen by intuition and then tested empirically for comparison. Through qualitative comparison, one can reduce the number of algorithms that need to be implemented by eliminating those algorithms that are likely to be weak combiners or to show poor generalization capability. This report surveys candidate algorithms that are likely to show good generalization performance for later empirical evaluation.

Contents

Figures

1. Introduction

The purpose of automatic target recognition (ATR) procedures is to identify targets quickly and precisely from a moving vehicle such as a tank or helicopter. The images are created from forward-looking infrared (FLIR) data. These data are difficult to screen because of the enormous number of variables involved. An image of a tank, say, could have been taken head on, from a side view, or from many other different angles. The image also tends to look different at different times of the day and in different environments. Further, it is unlikely that an enemy tank will be placed in full view for the sake of a good FLIR image. Thus, the ATR must also be able to identify a target that is partially obscured. Although all these factors may seem disheartening, two ATR procedures have been developed and tested with some success at the Army Research Laboratory (ARL). We first outline the four stages that are common to both of these algorithms [1] and then give a detailed description of each process (sect. 3).

The first stage of the algorithms screens an image for any potential targets and isolates regions of interest. Some of these potential targets turn out to be objects such as shrubbery. These types of false targets are called *clutter*. The second stage eliminates all the clutter from the set of potential targets. The third stage rank orders the potential target into a class by some quantitative measurement. The final stage chooses a class for a target using the information in the third stage. This stage, called classification, is the primary focus of this report.

Following a mathematical statement of the problem of classification (sect. 2), we provide an overview (sect. 3) of the two classifier schemes currently being used at ARL. Since the aim of this report is to choose a class given several classifiers, in section 4 we discuss how to create many classifiers given a certain training data set. Section 5 reviews (with examples) three techniques for combining classifiers already used at ARL, giving pros and cons. Finally, we outline several voting schemes that may be used to choose a target class.

2. Statement of Problem

The physical problem in ATR is to take a given target image $\mathbf{x}$ and assign this image to a class q of possible targets. When several classifiers are being used to solve a target recognition problem, the results between classifiers sometimes differ. For example, one classifier may identify tanks well, while another recognizes jeeps well. Intuition tells us that we should be able to combine the results of both classifiers and improve the overall performance. Mathematically, we can state the problem of combining classifiers as follows:

We are given an input vector $\mathbf{x} \in I\!R^n$, a family of classifiers $\mathcal{C} = \{\mathcal{C}_i\}_{i=1}^K$ and Q distinct classes. Each classifier $\mathcal{C}_i \in \mathcal{C}$ maps $\mathbf{x}$ to a vector $\mathbf{y}_i \in I\!R^Q$. An individual component of $\mathbf{y}_i$, $y_{i,q}$, represents the estimated probability computed by classifier $\mathcal{C}_i$ that $\mathbf{x}$ belongs to class q.

We are looking for a map $\mathcal{F}$ that assigns the matrix $\{\mathbf{y}_i\}_{i=1}^K$ to a class q, $1 \leq q \leq Q$. Essentially, we want to combine the information given by all the output vectors $\mathbf{y}_i$ and choose an appropriate target class for the vector $\mathbf{x}$.

3. The Classifiers ATR_LVQ and ATR_MNN

Currently there are two classifiers at ARL available for use by a combining algorithm, ATR_LVQ and ATR_MNN.

The first of these, ATR_LVQ, classifies targets by a matching system. The target image is input and matched in a codebook that uses mean square error as the matching metric. The process used by this classifier can be separated into four stages [2]. The first stage clips all the background clutter surrounding the target. After this stage, all the target chips are different sizes, so the next stage enlarges all the chips to one fixed size. The third stage decomposes the target chip into four subbands using wavelet decomposition. The aim of this step is to extract specific features of the target that are less complex than the entire image itself. For this reason, the subband vectors are sometimes called *feature vectors.* The final stage attempts to match up the features to a set of vector quantization (VQ) codebooks. A VQ codebook is a set of subband vectors called code vectors created for a specific target. The system does the matching by choosing the code vectors that minimize the distance to the input target vector.

In learning vector quantization (LVQ), the system constructs the codebook by updating each code vector through checking its performance on training data. For example, suppose there are several incorrect code vectors nearer to the target than the correct code vector. Updating the code vectors consists of the incorrect code vectors being "pushed" back and the correct one "pulled" closer. The specific details for the algorithm are provided elsewhere [3].

The second classifier available at ARL, ATR_MNN, combines the output of several neural networks, called *expert networks,* that are trained on specific local features of an image. The target image is broken into smaller blocks and searched for edges and corners. The feature extraction technique used is called the *interest operator.* Because target edges can be of high or low resolution and because the edge resolution is significant for classification, the interest operator is applied at several resolutions. (Wang et al [4] give details on the ATR_MNN process.) All the feature information from each receptive field is input into its own expert neural network. Finally, the output of all the expert networks is combined through stacked generalization (see sect. 5.2).

If two classifiers are used, only two voting methods are possible: a majority or anti-majority rule. If more voting options are needed, we need more than two classifiers.

4. Creating Multiple Classifiers

Jointly training a large number of classifiers and a classifier combination algorithm allows each new classifier to be trained to perform best on those images that the previous classifiers could not classify. This method can avoid the problem of having multiple classifiers with nearly identical performance characteristics. In this section, we discuss the general idea of joint training of multiple classifiers and describe one particular algorithm for implementing it. The discussion is based on work published elsewhere [5–7]; the theoretical background and support for this idea are given by Schapire [7] and Kearns and Valiant [8].

The proposed joint training algorithm has the following two steps.

1. Create several cheap classifiers.

The requirement for each of these classifiers is that the error rate of each be slightly less than 50 percent. Basically, each classifier is asked to do a little better than random guessing. Because of their high error rates, these classifiers are called *weak classifiers* [5]. We do not force the individual classifiers to be highly accurate, because we want them to be easy and cheap to create.

2. Combine the classifier outputs.

The end result of the combining step should be that the error of the combined output is considerably lower than that of the individual classifiers.

In section 5, we discuss some techniques for combining classifier outputs that have appeared in the literature. In this section, we concentrate on the first step, creating several classifiers. Although there are probably many ways to create weak classifiers, we describe one particular method, called *boosting*, which is very clearly explained by Drucker et al [6]. We follow their explanation in the step-by-step example given below, in which we create three classifiers. We are initially given a training data set of size N.

Step 1: Randomly choose a training set of size $N_1 < N$ to train Network 1.

Step 2: Create a training data set for Network 2 that has a 50-percent error rate in Network 1.

Let the vector $\mathbf{x} = (x_i)_{i=1}^{N-N_1}$ represent the entire training data set minus the elements used to train Network 1. Flip a coin. If it lands on heads, we begin putting patterns from $\mathbf{x}$ into Network 1 until there is a pattern x_j that is misclassified. The pattern x_j is put into the training data set of Network 2. If the coin lands on tails, we put the patterns into Network 1 until there is a pattern x_k that is classified correctly. The pattern x_k is placed into the training data set of Network 2.

This procedure is repeated through the list of patterns in $\mathbf{x}$ until Network 2 has a training data set of size N_1. The training data set for Network 2 will have an error rate of 50 percent in Network 1.

Step 3: Create a training data set for Network 3.

Collect the patterns in the original training data set minus the training data sets used for Networks 1 and 2. Pass these patterns through both Networks 1 and 2. If there is a pattern for which the two networks disagree, place this pattern in the training data set of Network 3. This process is repeated until there are N_1 patterns in the training data set of Network 3.

The obvious disadvantage of this method is the large amount of training data needed to fulfill the requirements of the algorithm. Drucker et al [6] propose another method, which recycles training data and consequently uses fewer data. Experimentally, however, it was seen that boosting tends to work better for large data sets, and a single network worked better for small training data sets.

5. Review of Combining Methods

We examine three different methods used to combine classifier outputs, which we have applied to the ATR_LVQ and ATR_MNN classifiers (sect. 3).

5.1 Averaged Bayesian Classifier

The averaged Bayesian classifier estimates the posterior probability of the combined output to be

$$\mathbf{y} = \frac{1}{K}\sum_{j=1}^{K}\mathbf{y}_j.$$

To obtain the final class for target $\mathbf{x}$, we choose the index of the maximal element in $\mathbf{y}$.

Example 1:

Consider two classifiers, $\mathcal{C}_1$ and $\mathcal{C}_2$, with three classes. Let

$$\mathbf{y}_1 = \begin{bmatrix} 0.2 \\ 0.5 \\ 0.3 \end{bmatrix} \quad \text{and} \quad \mathbf{y}_2 = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.8 \end{bmatrix}.$$

Then

$$\begin{aligned} \mathbf{y} &= \frac{1}{2}\left(\begin{bmatrix} 0.2 \\ 0.5 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.1 \\ 0.8 \end{bmatrix}\right) \\ &= \frac{1}{2}\begin{bmatrix} 0.3 \\ 0.6 \\ 1.1 \end{bmatrix} = \begin{bmatrix} 0.15 \\ 0.3 \\ 0.55 \end{bmatrix}. \end{aligned}$$

Since 0.55 is the maximal element in $\mathbf{y}$, $\mathbf{x}$ is assigned to Class 3.

Pros:

The main advantage to this method is that we are combining all the output in a decision. For example, if we compare the vectors

$$\begin{bmatrix} 0.8 \\ 0.1 \\ 0.1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0.5 \\ 0.4 \\ 0.1 \end{bmatrix},$$

majority vote tells us that Class 1 is the winner for both of these vectors. But we lose the information telling us how much the winner won by. This information is preserved in the averaged Bayesian classifier.

Cons:

This method will work well only if the estimated probabilities are good enough. The method does not account for interclassifier and intraclassifier biases.

5.2 Stacked Generalization

Stacked generalization uses a neural network to combine the outputs $(\mathbf{y}_i)_{i=1}^{K}$. The outputs from each classifier are fed to a neural network, as shown in figure 1. The final result of the stacked generalizer is obtained by the composition of a nonlinear function Φ and a linear combination of the vectors $(\mathbf{y}_i)_{i=1}^{K}$. The process in stacked generalization works as follows. There are a certain number of parameters (weights) that need to be estimated for the neural net in the stacked generalizer. The vectors $(\mathbf{y}_i)_{i=1}^{K}$ are computed for an input for which the ground truth is known. The error is determined and the weights of the stacked generalizer are adjusted. This procedure is done for a large amount of data until the weights tend to settle to a constant value for any new input.

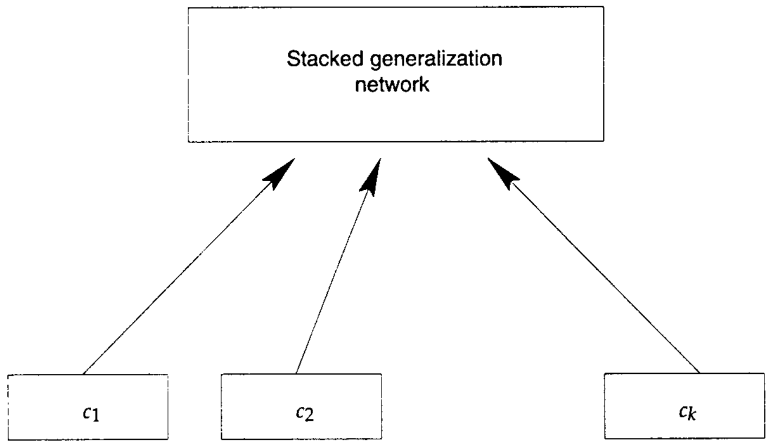

Figure 1. Stacked generalization process.

Pros:

Experimentally, the results of the stacked generalization method are extremely successful. Compared to the other two methods, it produces the highest number of correct classifications. Theoretically, the method also accounts for intra- and inter-classifier biases.

Cons:

This method is computationally complex. It is difficult to gather the amount of data needed to properly train the weights. The whole process of stacked generalization is so complicated that it is nontrivial to understand exactly what occurred within the structure of the neural network.

5.3 Quality Based Cascade Classifier

Let $z_i = \max_q\{y_{i,q}\}$ and $\hat{z}_i = \max_q\{y_{i,q} \backslash z_i\}$. Define the *certainty* for classifier $\mathcal{C}_i$ to be

$$c_i = z_i - \hat{z}_i.$$

It was found experimentally that if the certainty of a classifier was high, the probability that the classifier had made the correct choice was also high.

The cascading method chooses a threshold certainty λ_i for each classifier $\mathcal{C}_i, i = 1, \dots K$. $\mathcal{C}_1$ is executed, and the certainty c_1 is computed. If $c_1 \geq \lambda_1$, then the target is considered to be correctly classified by $\mathcal{C}_1$. If $c_1 < \lambda_1$, then the target is passed to the next classifier, $\mathcal{C}_2$, where the process is repeated. In this manner, the inputs are "cascaded" down a chain.

Example 2:

Consider the situation in Example 1. Let $\lambda_1 = \lambda_2 = 0.5$. For $\mathbf{y}_1$, $c_1 = 0.5 - 0.3 = 0.2$. Since $c_1 < 0.5$, the input is passed to $\mathcal{C}_2$.

For $\mathcal{C}_2$, $c_2 = 0.8 - 0.1 = 0.7 > 0.5$. Since $c_2 > 0.5$, we classify this target as Class 3.

Pros:

As opposed to the Bayesian classifier and stacked generalization, the cascade classifier does not execute each $\mathcal{C}_i$ for every target, so the computational complexity is lower. In addition, not all the classifiers are weighted equally, as in the Bayesian classifier, and so the cascade classifier can account for interclassifier biases.

Cons:

If a classifier misclassifies a target with high certainty, we have no way to use the other classifiers to cross check the error. Another problem is that we must be able to rank our classifiers, which may be difficult. Moreover, there is no set way to choose the λ_i's.

6. Other Voting Methods

We are all familiar with the concept of voting, particularly the majority vote. Majority vote performs well (in terms of probability of correct classification) when there are only two candidates (for us, two classes). When there are three candidates, however, the situation becomes complex. There is a nice example of a three-candidate case in the first few pages of Saari's *The Geometry of Voting* [9], which describes a case where the majority winner of a three-candidate race did not correctly reflect the preferences of the voters. Through several examples in the first chapter of his book, Saari shows that an election outcome depends more on the election process than on the voters' desires.

From our point of view, we should choose a voting method to combine the classifier outputs after some mathematical examination. This section focuses on several voting procedures and how appropriate they are for the purposes of target recognition.

6.1 The Borda Count

The Borda count (BC) is a method proposed by the French mathematician J. C. Borda in 1770 [9]. The BC assigns $Q-i$ points to the ith ranked class for $i = 1, \ldots Q$. The numbers of points assigned to each class are summed, and a winner is chosen by the highest total. The BC is applied in the following example.

Example 3:

Assume we have three classifiers with four classes with the output vectors given to be

$$y_1 = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.1 \\ 0.4 \end{bmatrix}, \; y_2 = \begin{bmatrix} 0.2 \\ 0.1 \\ 0.4 \\ 0.3 \end{bmatrix}, \; y_3 = \begin{bmatrix} 0.5 \\ 0.2 \\ 0 \\ 0.3 \end{bmatrix}.$$

We define the symbol $\succ$ to mean "preferred over." Then we can write the outcomes of the classifiers as follows:

- Classifier 1: Class 4 $\succ$ Class 2 $\succ$ Class 1 $\succ$ Class 3
- Classifier 2: Class 3 $\succ$ Class 4 $\succ$ Class 1 $\succ$ Class 2
- Classifier 3: Class 1 $\succ$ Class 4 $\succ$ Class 2 $\succ$ Class 3

Then we assign points to each class using the BC, as shown in figure 2. We see that the BC winner for this example is Class 4. So our target is classified as Class 4.

Ho et al [10] used the BC to fuse classifiers. The main objection to the Borda count is the equal weighting of each classifier. A similar problem arises in the averaged Bayesian classifier. *Approval voting* may be a way to get around this situation.

6.2 Approval Voting

Approval voting was invented independently by several people in the late 1970's [9]. Its aim is to give a voter a chance to examine each candidate and vote "yes" or "no." For example, suppose there were three candidates for the presidential chair of the American Mathematical Society (AMS), Michael Jones, Franklin Mendivil, and Machiel van Frankenhuyser. Say I prefer Michael and Franklin equally but have no opinion of Machiel since I do not know him well. Then I would assign a 1 for both Mike and Franklin and assign a 0 for Machiel. This method of voting is similar to the idea of *class reduction* discussed by Ho et al [10]. Basically, we are reducing the total amount of data by throwing out certain elements.

Approval voting may be implemented for classifier fusion in the following manner. We first choose a threshold value λ_i for each classifier C_i, similar to cascading. If the estimated posterior probability $y_{i,q}$ for classifier C_i and class q is greater than or equal to λ_i, we assign that class a vote of 1. If $y_{i,q} < \lambda_i$, we assign the class a vote of 0. Finally we sum the voting vectors, and the largest value determines the classification.

	Class 1	Class 2	Class 3	Class 4
Classifier 1	(4 – 3) = 1	(4 – 2) = 2	(4 – 4) = 0	(4 – 1) = 3
Classifier 2	(4 – 3) = 1	(4 – 4) = 0	(4 – 1) = 3	(4 – 2) = 2
Classifier 3	(4 – 1) = 3	(4 – 3) = 1	(4 – 4) = 0	(4 – 2) = 2
Totals	5	3	3	7

Figure 2. Borda count totals.

Example 4:

Let there be three classifiers with four classes. The classifier vectors are given as

$$y_1 = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{bmatrix}, \ y_2 = \begin{bmatrix} 0.4 \\ 0.5 \\ 0.1 \\ 0 \end{bmatrix}, \ \text{and} \ y_3 = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.3 \\ 0.1 \end{bmatrix}.$$

Choose threshold $\lambda_1 = 0.2, \lambda_2 = 0.4$, and $\lambda_3 = 0.3$. Then the voting vectors are

$$z_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \ z_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \ \text{and} \ z_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

The sum of the z_i's yields the vector

$$\begin{bmatrix} 2 \\ 3 \\ 1 \\ 1 \end{bmatrix},$$

which produces Class 2 as the winning class.

One point mentioned by Saari [9] is that approval voting does not provide the option of strict ranking. For example, suppose that I preferred Mike to Franklin, as opposed to having an equal preference for both. I would want to give an order to my vote, ranking Mike the highest, Franklin second, and Machiel last. This voting vector is not defined in approval voting. We can also guess that with many classifiers, the possibility of a tie becomes more probable if approval voting is used. *Cumulative voting* may provide an alternative to this situation.

6.3 Cumulative Voting

Cumulative voting gives each voter a certain number of points to award each candidate. The way to implement cumulative voting without weighting each classifier equally is again to choose threshold values. We choose a threshold λ_i for each C_i. Once again, we "throw out the bad apples" by assigning the value 0 to everything below the threshold. But what is different is that we will rank order each element of each classifier vector before throwing out the bad elements. Then we sum the resulting voting vectors and let the largest value determine the class.

Example 5:

Consider the vectors and threshold values in the previous example. Rank ordering yields the vectors

$$z_1 = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 4 \end{bmatrix}, \; z_2 = \begin{bmatrix} 3 \\ 4 \\ 2 \\ 1 \end{bmatrix}, \; \text{and} \; z_3 = \begin{bmatrix} 2 \\ 4 \\ 3 \\ 1 \end{bmatrix}.$$

Throwing out the elements that fall below the threshold values yields the new voting vectors:

$$\hat{z}_1 = \begin{bmatrix} 3 \\ 2 \\ 0 \\ 4 \end{bmatrix}, \; \hat{z}_2 = \begin{bmatrix} 3 \\ 4 \\ 0 \\ 0 \end{bmatrix}, \; \text{and} \; \hat{z}_3 = \begin{bmatrix} 0 \\ 4 \\ 3 \\ 0 \end{bmatrix}.$$

The sum of the $\hat{z}_i$'s is

$$\begin{bmatrix} 6 \\ 10 \\ 3 \\ 4 \end{bmatrix},$$

and so Class 2 is once again our winning class. The immediate advantage over approval voting is that cumulative voting allows for all vectors: the strict ranking voting vector, a majority type of voting vector, and an equal preference voting vector.

The weights in the ranking need not be given in order. For example, if I prefer the first class in the first classifier to all the other classes, I may assign all four points to that class and nothing to the rest. Thus I can have a staggered set of threshold values to assign the points. As with cumulative and approval voting, there is no theoretical foundation for choosing threshold values for the cascade classifier. There is probably no set way to choose such thresholds.

7. Conclusions

The large body of literature on classifier combining methods testifies to its usefulness in achieving high classification performance on real-world problems. Multiple classifiers allow each individual classifier to solve a subset of the problem, avoiding the overcomplexity that leads to poor generalization performance on unseen data. The multiple classifier method frequently requires less training time for the system because of this reduced complexity. This report surveys some techniques that show promise on the particular problem of target classification in outdoor imagery.

References

1. B. Bhanu, "Automatic target recognition: State of the art survey," *IEEE Trans. Aerosp. Electron. Syst.* **AES-22**, no. 4, 364–379, July 1986.

2. Lin-Cheng Wang, Lipchen A. Chan, Nasser M. Nasrabadi, and Sandor Z. Der, "Combination of two learning algorithms for automatic target recognition," *IEEE Int. Conf. Image Processing 1997*, Santa Barbara, CA, 26–30 October 1997.

3. Lin-Cheng Wang, Sandor Z. Der, and Nasser M. Nasrabadi, "A committee of networks classifier with multi-resolution feature extraction for automatic target recognition," *IEEE Int. Conf. Neural Networks 1997*, Houston, TX, 9–12 June 1997.

4. Lin-Cheng Wang, Sandor Z. Der, Syed A. Rizvi, and Nasser M. Nasrabadi, "Automatic target recognition using a neural network with directional variance features," *Proc. IS&T/SPIE Symposium on Electronic Imaging: Science and Technology*, 8–14 February 1997.

5. Chuanyi Ji and Sheng Ma, "Combinations of weak classifiers," *IEEE Trans. Neural Networks* **8**, no. 1, 32–42, January 1997.

6. Harris Drucker, Corinna Cortes, L. D. Jackel, Yann LeCun, and Vladimir Vapnik, "Boosting and other ensemble methods," *Neural Comput.* **6**, 1289–1301, 1994.

7. R. Schapire, "The strength of weak learnability," *Machine Learn.* **5**, no. 2, 197–227, 1990.

8. M. Kearns and L. G. Valiant, "Cryptographic limitations on learning Boolean formulae and finite automata," *Proc. Nineteenth Ann. ACM Symp. Theory Computing*, 443–444, 1989.

9. Donald G. Saari, *The Geometry of Voting*, Springer-Verlag, 1994.

10. Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Machine Intell.* **16**, no. 1, 66–75, January 1994.

Distribution

Admnstr
Defns Techl Info Ctr
Attn DTIC-OCP
8725 John J Kingman Rd Ste 0944
FT Belvoir VA 22060-6218

Ofc of the Dir Rsrch and Engrg
Attn R Menz
Pentagon Rm 3E1089
Washington DC 20301-3080

Ofc of the Secy of Defns
Attn ODDRE (R&AT) G Singley
Attn ODDRE (R&AT) S Gontarek
The Pentagon
Washington DC 20301-3080

OSD
Attn OUSD(A&T)/ODDDR&E(R) R Tru
Washington DC 20301-7100

CECOM
Attn PM GPS COL S Young
FT Monmouth NJ 07703

CECOM RDEC Elect System Div Dir
Attn J Niemela
FT Monmouth NJ 07703

CECOM Sp & Terrestrial Commctn Div
Attn AMSEL-RD-ST-MC-M H Soicher
FT Monmouth NJ 07703-5203

Dir of Assessment and Eval
Attn SARD-ZD H K Fallin Jr
103 Army Pentagon Rm 2E673
Washington DC 20301-0163

Hdqtrs Dept of the Army
Attn DAMO-FDT D Schmidt
400 Army Pentagon Rm 3C514
Washington DC 20301-0460

MICOM RDEC
Attn AMSMI-RD W C McCorkle
Redstone Arsenal AL 35898-5240

US Army Avn Rsrch, Dev, & Engrg Ctr
Attn T L House
4300 Goodfellow Blvd
St Louis MO 63120-1798

US Army CECOM Rsrch, Dev, & Engrg Ctr
Attn R F Giordano
FT Monmouth NJ 07703-5201

US Army Edgewood Rsrch, Dev, & Engrg Ctr
Attn SCBRD-TD J Vervier
Aberdeen Proving Ground MD 21010-5423

US Army Info Sys Engrg Cmnd
Attn ASQB-OTD F Jenia
FT Huachuca AZ 85613-5300

US Army Materiel Sys Analysis Agency
Attn AMXSY-D J McCarthy
Aberdeen Proving Ground MD 21005-5071

US Army Matl Cmnd
Dpty CG for RDE Hdqtrs
Attn AMCRD BG Beauchamp
5001 Eisenhower Ave
Alexandria VA 22333-0001

US Army Matl Cmnd
Prin Dpty for Acquisition Hdqrts
Attn AMCDCG-A D Adams
5001 Eisenhower Ave
Alexandria VA 22333-0001

US Army Matl Cmnd
Prin Dpty for Techlgy Hdqrts
Attn AMCDCG-T M Fisette
5001 Eisenhower Ave
Alexandria VA 22333-0001

US Army Natick Rsrch, Dev, & Engrg Ctr
Acting Techl Dir
Attn SSCNC-T P Brandler
Natick MA 01760-5002

US Army Rsrch Ofc
Attn G Iafrate
4300 S Miami Blvd
Research Triangle Park NC 27709

US Army Simulation, Train, & Instrmntn Cmnd
Attn J Stahl
12350 Research Parkway
Orlando FL 32826-3726

Distribution

US Army Tank-Automtv & Armaments Cmnd
Attn AMSTA-AR-TD C Spinelli
Bldg 1
Picatinny Arsenal NJ 07806-5000

US Army Tank-Automtv Cmnd
Rsrch, Dev, & Engrg Ctr
Attn AMSTA-TA J Chapin
Warren MI 48397-5000

US Army Test & Eval Cmnd
Attn R G Pollard III
Aberdeen Proving Ground MD 21005-5055

US Army Train & Doctrine Cmnd
Battle Lab Integration & Techl Dirctrt
Attn ATCD-B J A Klevecz
FT Monroe VA 23651-5850

US Military Academy
Dept of Mathematical Sci
Attn D Thomas
Attn MAJ D Engen
West Point NY 10996

USAASA
Attn MOAS-AI W Parron
9325 Gunston Rd Ste N319
FT Belvoir VA 22060-5582

Nav Surface Warfare Ctr
Attn Code B07 J Pennella
17320 Dahlgren Rd Bldg 1470 Rm 1101
Dahlgren VA 22448-5100

GPS Joint Prog Ofc Dir
Attn COL J Clay
2435 Vela Way Ste 1613
Los Angeles AFB CA 90245-5500

DARPA
Attn B Kaspar
Attn L Stotts
3701 N Fairfax Dr
Arlington VA 22203-1714

ARL Electromag Group
Attn Campus Mail Code F0250 A Tucker
University of Texas
Austin TX 78712

Dir for MANPRINT
Ofc of the Deputy Chief of Staff for Prsnnl
Attn J Hiller
The Pentagon Rm 2C733
Washington DC 20301-0300

US Army Rsrch Lab
Attn AMSRL-CI-LL Techl Lib (3 copies)
Attn AMSRL-CS-AL-TA Mail & Records Mgmt
Attn AMSRL-CS-AL-TP Techl Pub (3 copies)
Attn AMSRL-SE J M Miller
Attn AMSRL-SE-E J Pellegrino
Attn AMSRL-SE-SE D Nguyen
Attn AMSRL-SE-SE J Phillips
Attn AMSRL-SE-SE L Bennett
Attn AMSRL-SE-SE M Vrabel
Attn AMSRL-SE-SE N Nasrabadi
Attn AMSRL-SE-SE P Rauss
Attn AMSRL-SE-SE S Der (5 copies)
Attn AMSRL-SE-SI T Kipp
Attn AMSRL-SE-SR D Rodkey
Attn AMSRL-SE-SR D Rosario
Attn AMSRL-SE-SR G Stolovy
Attn AMSRL-SE-SR J Dammann
Adelphi MD 20783-1197

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 1998	**3. REPORT TYPE AND DATES COVERED** Final, from April 1997 to August 1997

4. TITLE AND SUBTITLE Voting Techniques for Combining Multiple Classifiers	**5. FUNDING NUMBERS** PE: 61102A
6. AUTHOR(S) Diana Thomas (U.S. Military Academy) and Sandor Z. Der (ARL)	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory Attn: AMSRL-SE-SE (sder@arl.mil) 2800 Powder Mill Road Adelphi, MD 20783-1197	**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-1549
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory 2800 Powder Mill Road Adelphi, MD 20783-1197	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES
AMS code: 611102.305
ARL PR: 7NE0M1

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

Algorithms for decision fusion are surveyed and qualitatively compared for the problem of classification of targets. The methods assume that a number of imperfect classifiers are available, and that these classifiers have enough statistical independence that significant improvement can be made by good combination algorithms. Optimal solutions for this problem require an exact statistical model of the classifiers and the decision space, which are rarely available for real-world problems. Consequently, algorithms must be chosen by intuition and then tested empirically for comparison. Through qualitative comparison, one can reduce the number of algorithms that need to be implemented by eliminating those algorithms that are likely to be weak combiners or to show poor generalization capability. This report surveys candidate algorithms that are likely to show good generalization performance for later empirical evaluation.

14. SUBJECT TERMS ATR, classifier fusion, voting methods	**15. NUMBER OF PAGES** 24
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102